AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Currently Amended) A method of generating a software program executable binary file, the method comprising:

accessing, by a processing device, a first file for a first module, wherein the first file includes ~~including~~ source code ~~therein~~;

accessing, by the processing device, a second file for a second module, wherein the second file includes ~~including~~ machine-executable object code ~~therein~~ and ~~further including~~ object file summary information; and

generating, by the processing device, the executable binary file from at least the first and second files, wherein the object file summary information is used in optimizing the executable binary file generated,

wherein the object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table,[[and]] wherein the SIR includes information relating to symbols accessed by a procedure in the object code, and wherein the extension to the linker symbol table includes a flag indicating whether the procedure exposes a memory address by storing the memory address in a location accessible outside the procedure

~~wherein the object file summary information is used in optimizing the executable binary file generated~~.

2. (Original) The method of claim 1, further comprising disambiguating memory accesses otherwise considered aliased using the object file summary information.

3. (Cancelled)

4. (Cancelled)

5. (Previously Presented) The method of claim 1, wherein the SIR includes a summary symbol table.

6. (Currently Amended) The method of claim 5, wherein the summary symbol table includes global and static symbols accessed in [[a]]the procedure, formal parameters of the procedure, a return location for the procedure, and other procedures called by the procedure.

7. (Previously Presented) The method of claim 6, wherein a symbol is referenced in the summary symbol table by using an associated summary symbol identifier (SYMID).

8. (Cancelled)

9. (Original) The method of claim 5, wherein the SIR uses an operator for memory referencing.

10. (Previously Presented) The method of claim 5, wherein the SIR uses an operator to adjust an address expression by an offset.

11. (Original) The method of claim 5, wherein the SIR uses an operator to take an address of a function or variable.

12. (Original) The method of claim 5, wherein the SIR uses an operator to merge pointer values from different control flow paths.

13. (Original) The method of claim 5, wherein the SIR uses an operator to represent direct procedure calls.

14. (Original) The method of claim 5, wherein the SIR uses an operator to represent indirect procedure calls.

15. (Original) The method of claim 5, wherein the SIR uses a no-operation type operator to discard values.

16. (Original) The method of claim 5, wherein the SIR includes a control data structure comprising a link field for each procedure that points to an SIR block of a next procedure.

17. (Original) The method of claim 5, wherein the SIR includes a control data structure comprising a table having links to an SIR block for each procedure.

18. (Original) The method of claim 1, further comprising determining variables modified by and referenced by function calls in the object code using the object file summary information.

19. (Cancelled)

20. (Currently Amended) The method of claim 18, wherein the extension to the linker symbol table further includes a ~~first~~second flag indicative of whether [[a]]the procedure modifies non-local variables and a ~~second~~third flag indicative of whether the procedure references non-local variables.

21. (Currently Amended) The method of claim 20, wherein the ~~extension to the linker symbol table includes a~~ second flag is indicative of whether the procedure modifies global/static variables~~ excluding callees and a third flag indicative of whether the procedure references non-local variables excluding callees~~.

22. (Currently Amended) The method of claim 18, wherein the ~~per-procedure summary data~~SIR comprises a linked list of entries corresponding to symbols directly modified or referenced in a procedure.

23. (Original) The method of claim 22, wherein each entry comprises a linker identifier of a corresponding symbol and flags indicative of whether that symbol is modified or referenced.

24. (Original) The method of claim 1, wherein the second file comprises a load module that is a shared library of procedures.

25. (Currently Amended) The method of claim 1, wherein multiple files including object code are accessed and used in ~~compiling the program~~generating the software program executable binary file.

26. (Currently Amended) A system for generating a software program executable file, the system comprising:

a processing device configured to execute computer-readable program code;

a memory system configured to store the computer-readable program code and data;

a source file for a first module comprising source code stored by the memory system;

an object file for a second module including ~~computer-readable program~~ machine-executable object code and object file summary information; and

a translator comprising computer-readable program code stored by the memory system, wherein the computer-readable program code of the translator is configured to access at least the source and object files and to generate the software program executable file ~~of the program~~ therefrom while using the object file summary information to optimize the generated software program executable file,

wherein the object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table,[[and]] wherein the SIR includes information relating to symbols accessed by a procedure in the object code, and wherein the extension to the linker symbol table includes a flag indicating whether the procedure exposes a memory address by storing the memory address in a location accessible outside the procedure

~~wherein the object file summary information is used in optimizing the executable file generated~~.

27. (Currently Amended) The system of claim 26, further comprising a points-to analyzer ~~that uses~~ configured to use the object file summary information to disambiguate memory accesses otherwise considered aliased.

28. (Currently Amended) The system of claim 26, further comprising a module ~~that uses~~ configured to use the object file summary information to determine variables modified by and referenced by function calls in the object file.

29. (Currently Amended) The system of claim 26, wherein the translator comprises:

a compiler configured to translate the source ~~files~~file into an intermediate ~~files~~file; and

a linker configured to access the object file summary information and communicate information to the compiler relevant to optimizing compilation of the software program.

30. (Original) The system of claim 29, wherein the translator further comprises a feedback provider that provides a communications interface between the compiler and the linker.

31. (Currently Amended) A computer-readable storage medium storing code that upon execution by a processing device causes the processing device to:

access a first file for a first module, wherein the first file includes source code;

access a second file for a second module, wherein the second file includes machine-executable object code and object file summary information; and

generate an executable binary file from at least the first and second files, wherein the object file summary information is used in optimizing the executable binary file generated,

wherein the object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table, wherein the SIR includes information relating to symbols accessed by a procedure in the object code, and wherein the extension to the linker symbol table includes a flag indicating whether the procedure exposes a memory address by storing the memory address in a location accessible outside the procedure~~an object file of a computer programming module, the computer-readable storage medium comprising:~~

~~computer-readable object code for the module; and~~

~~computer-readable object file summary information including a summary intermediate representation (SIR) and an extension to a linker symbol table for use by a compiler in optimizing executable code including the module~~.

32. (Previously Presented) The computer-readable storage medium of claim 31, wherein the SIR includes a summary symbol table.

33. (Currently Amended) The computer-readable storage medium of claim 32, wherein the summary symbol table includes global and static symbols accessed in the ~~module~~procedure, formal parameters of the ~~module~~procedure, a return location for the ~~module~~procedure, and other procedures called by the ~~module~~procedure.

34. (Currently Amended) The computer-readable storage medium of claim 31, wherein the SIR uses a plurality of operators from a group of operators including an operator for memory referencing, an operator to adjust the address expression by an offset, an operator to take an address of a function or variable, an operator to merge pointer values from different control flow paths, an operator to represent direct procedure calls, an operator to represent indirect procedure calls, [[.]]and a no-operation type operator to discard values.

35. (Cancelled)

36. (New) The method of claim 1, wherein the SIR further includes exposed pointer assignments, wherein each of the exposed pointer assignments is an assignment of a pointer that is visible to a caller of the procedure.

37. (New) The system of claim 26, wherein the SIR further includes exposed pointer assignments, wherein each of the exposed pointer assignments is an assignment of a pointer that is visible to a caller of the procedure.

38. (New) The computer readable storage medium of claim 31, wherein the SIR further includes exposed pointer assignments, wherein each of the exposed pointer assignments is an assignment of a pointer that is visible to a caller of the procedure.